CLAIMS

1. A runtime system for a global address space language for use with a plurality of processors or computers, the system comprising:

a directory of shared variables comprising a data structure for tracking shared variable information that is shared by a plurality of program threads; and

allocation and de-allocation routines for allocating and de-allocating shared variable entries in the directory of shared variables.

2. The runtime system of claim 1 wherein the allocation and de-allocation routines use pair-to-pair synchronization.

3. The runtime system of claim 1 wherein the runtime system is implemented on a distributed memory system and the directory of shared variables is stored in a private memory of each thread such that it is replicated across all of the threads.

4. The runtime system of claim 1 wherein the runtime system is implemented on a shared memory system and the directory of shared variables is stored in a shared memory shared by all threads.

5. The runtime system of claim 1 wherein the allocation and de-allocation routines are used for both statically and dynamically allocated data.

6. The runtime system of claim 1 wherein arrays that are dynamically allocated have affinity to a thread that called the allocation or de-allocation routine.

7. The runtime system of claim 1 wherein every thread has a handle for each shared variable that it accesses.

8. The runtime system of claim 7 wherein the entries in the directory of shared variables are accessed using the handle.

9. The runtime system of claim 7 wherein the handle comprises a partition index and a variable index.

10. The runtime system of claim 1 wherein each thread has exclusive write access rights to a partition of the directory of shared variables associated with the thread.

11. A runtime system that scales to a plurality of processors for a global address space language program having a plurality of threads that access memory in a global address space system, the system comprising:

a shared data directory that maintains shared data entries related to shared data structures that are shared by more than one of the plurality of threads; and

control structures to access, allocate and de-allocate the shared data structures through the shared data directory.

12. The runtime system of claim 11 wherein the plurality of processors operate as a shared memory machine.

13. The runtime system of claim 11 wherein the plurality of processor operate as a distributed memory machine.

14. The runtime system of claim 11 wherein the shared data structures have affinity to particular threads.

15. The runtime system of claim 11 wherein the shared data structures comprise shared scalar variables, objects, arrays or pointers.

16. The runtime system of claim 15 wherein a shared scalar variable is accessed by dereferencing a shared data directory partition for which the shared scalar variable has affinity.

17. The runtime system of claim 15 wherein a shared array has a shared data

directory partition that points to a control structure that points to the shared array.

18. The system of claim 15 wherein the runtime system allocates a control harness for a shared pointer when the shared pointer is declared by allocating a shared control block and a shared address structure.

19. The system of claim 15 wherein some of the shared pointers have shared targets and some of the shared pointers have private targets.

20. The system of claim 11 wherein entries to the shared data directory are allocated by an owning thread or, in a synchronized manner by all threads at the same time.

21. The runtime system of claim 11 comprising a handle that includes a partition index and a variable index that is used by the threads to access the shared variables.

22. The runtime system of claim 11 wherein the shared data directory includes a partition that is used to access all statically declared non-scalar variables.

23. The runtime system of claim 11 wherein each thread uses a mutually exclusive partition of the shared data directory.

24. A method of providing a scalable runtime system for a global address space language, the method comprising:

creating a directory of shared variables containing information concerning data shared by program threads for use by the threads in accessing the shared data; and

creating control structures to control allocation and de-allocation of the shared data.

25. The method of claim 24 wherein creating control structures comprises creating a plurality of control structures wherein each control structure controls the allocation and de-allocation of a particular type of shared data structure.

26. The method of claim 24 comprising operating the runtime system on a distributed memory machine.

27. The method of claim 26 wherein each thread contains a private copy of the directory of shared variables and a calling thread allocates an entry in its directory of shared variables and broadcasts an index of the entry to other threads.

28. The method of claim 26 wherein each thread has a private data control structure with a pointer to a shared memory fraction.

29. The method of claim 24 comprising operating the runtime system on a shared memory machine.

30. The method of claim 24 wherein a calling thread allocates space for a shared variable and inserts a handle in a partition in the directory of shared variables.

31. The method of claim 29 wherein the control structures are common such that any thread can access the common control structures.